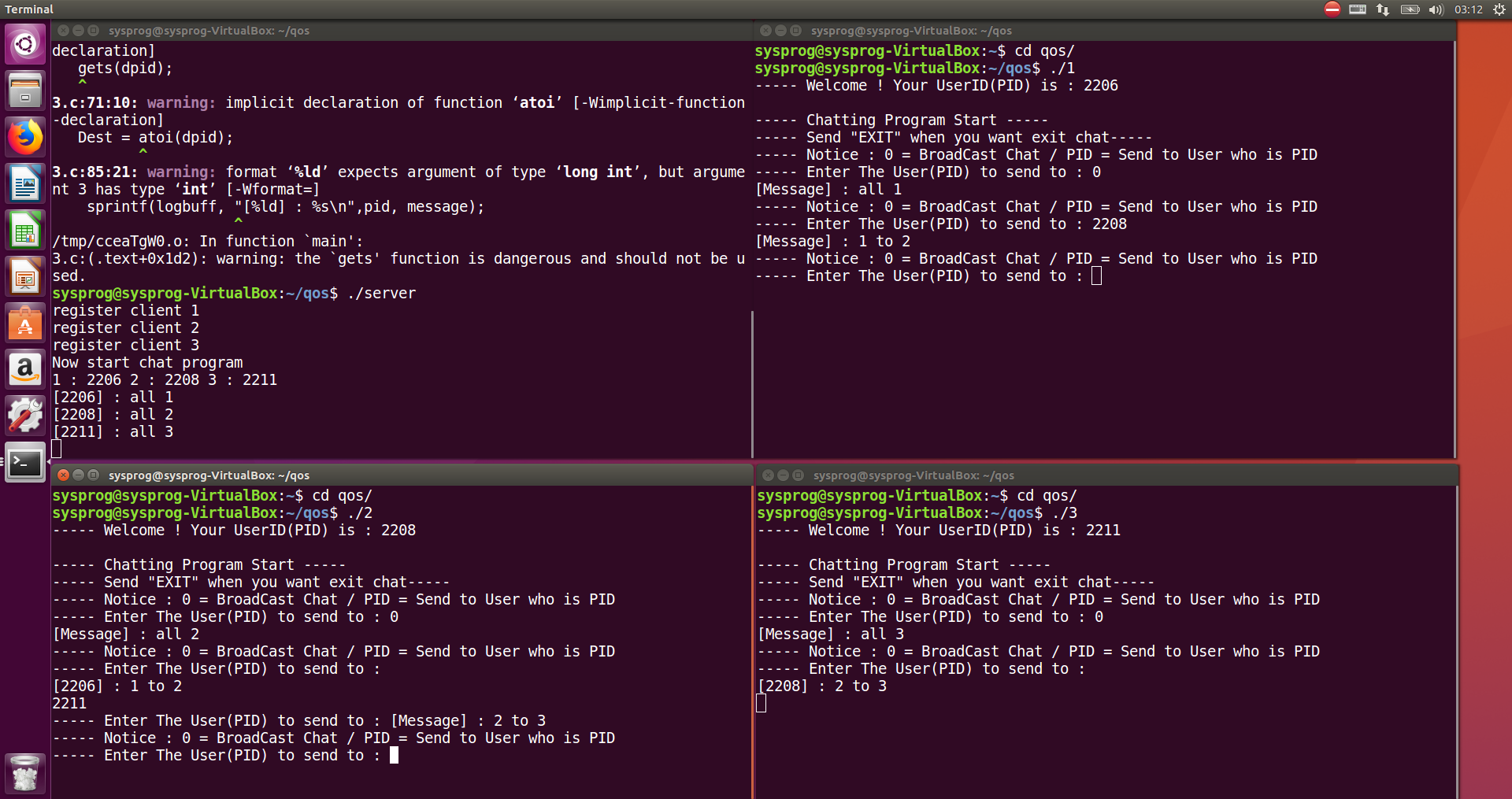
System Programming Project 7 IPC

2013011253 김기원

2014004248 강준규

보고서 결과물 : https://github.com/etual118/sysprogpj3

1. 실행결과



2. server.c

#define SHARE\_MEMORY\_KEY 4458

#define MESSAGE\_QUEUE\_CLIENT\_ONE 1818

#define MESSAGE\_QUEUE\_CLIENT\_TWO 1819

#define MESSAGE\_QUEUE\_CLIENT\_THREE 1820

#define CL\_SIZE 105000

#define SHARE\_TEMP\_KEY 4500

#define TEMP\_SIZE 1050

#define GLOBAL\_TYPE 100000

//우선 서로 사용하는 IPC 수단을 같게하기 위해서 key를 하드코딩 해줍니다. Server는 각 client와 통신하기 위해서 하나씩의 message queue를 생성합니다.

char \* logmem;

char \* tmpmem;

int SMID, SMID\_C;

int count = 0;

typedef struct {

long type;

long pid;

char mbuff[1024];

} Msq;

typedef struct {

int MQID;

long pid;

} Client;

Client client[4];

//아래는 스레드의 str\_rt으로 지정되기 위한 함수입니다. 각 클라이언트를 담당하기 위해서 클라이언트 개수만큼의 스레드가 선언됩니다.

void \*receiver1() {

Msq rcvq;

int msgSize = sizeof(rcvq) - sizeof(rcvq.type);

while (1) {

if (msgrcv(client[1].MQID, &rcvq, msgSize, client[1].pid, MSG\_EXCEPT) > 0) {

//수신 시 type과 pid가 같다면 자기 자신에게 보내는 메시지이거나 이미 서버가 분류하여 삽입해준 메시지이므로 서버가 관리하지않고 클라이언트가 알아서 처리할 수 있게 type과 pid가 같은 메시지는 읽지 않습니다. 이는 MSG\_EXCEPT 옵션을 사용함으로써 해결하였습니다.

if (!strcmp(rcvq.mbuff, "EXIT")) {

pthread\_exit(NULL);

}

//만일 EXIT메시지가 날아온다면 스레드를 종료합니다.

if (rcvq.type >= GLOBAL\_TYPE) {

printf("%s", tmpmem);

memcpy(logmem + (count % 100) \* 1050, tmpmem, TEMP\_SIZE);

count++;

}

//임시 메모리에 받아온 값을 출력 후, 로그에 저장합니다. 로그는 100개 까지 저장하기 위해서 순환형으로 돌아가게됩니다.

else {

if (rcvq.type == client[1].pid) {

msgsnd(client[1].MQID, &rcvq, msgSize, 0);

}

else if (rcvq.type == client[2].pid) {

msgsnd(client[2].MQID, &rcvq, msgSize, 0);

}

else if (rcvq.type == client[3].pid) {

msgsnd(client[3].MQID, &rcvq, msgSize, 0);

}

else

printf("fatal err\n");

}

}

}

//만일 전체 채팅이 아닌 클라이언트 간 채팅이라면 메시지를 다른 메시지큐로 옮겨줍니다.

}

//main 함수에서 잡다한 선언들은 설명에서 제외하겠습니다. 채팅 프로그램을 정상적으로 동작시키기 위해서는 우선 server를 먼저 실행한 후, 1번 2번 3번의 순으로 클라이언트를 열어줘야 합니다. 각 클라이언트는 메시지를 하나 날려서 자신의 pid를 알리고, 서버는 이를 저장해둡니다.

/

int main()

{

tmpmem = (char \*)shmat(SMID\_C, NULL, 0);

printf("register client 1\n");

msgrcv(client[1].MQID, &rcvq, msgSize, 0, 0);

client[1].pid = rcvq.pid;

printf("register client 2\n");

msgrcv(client[2].MQID, &rcvq, msgSize, 0, 0);

client[2].pid = rcvq.pid;

printf("register client 3\n");

msgrcv(client[3].MQID, &rcvq, msgSize, 0, 0);

client[3].pid = rcvq.pid;

pthread\_t tid[3];

pthread\_create(&tid[0], NULL, receiver1, NULL);

pthread\_create(&tid[1], NULL, receiver2, NULL);

pthread\_create(&tid[2], NULL, receiver3, NULL);

printf("Now start chat program\n1 : %ld 2 : %ld 3 : %ld\n", client[1].pid, client[2].pid, client[3].pid);

for (int i = 0; i < 3; i++)

pthread\_join(tid[i], NULL);

shmdt(logmem);

shmctl(SMID, IPC\_RMID, NULL);

printf("Now exit chat program\n");

return 0;

}

//전부 끝난다면 사용했던 IPC들을 정리해줍니다.

3. client.c

#define MESSAGE\_QUEUE\_KEY XXXX

//각 클라이언트는 개별적으로 server에 등록된 키를 적어야 합니다.

typedef struct{

long type;

long pid;

char mbuff[1024];

} Msq;

Msq sndq,rcvq;

int SMID\_C, SMID, MQID, pid, Dest, msqType;

int msgSize = sizeof(sndq) - sizeof(sndq.type);

int rcvSize = 0;

rcv 함수는 스레드로 선언되어 rcv 역할을 전담합니다. Pid와 같은 type인 메시지들만 수신하여 출력합니다.

void \*rcvT()

{

while(1)

{

if(msgrcv(MQID, &rcvq, msgSize, pid, IPC\_NOWAIT) > 0)

{

//printf("receive type is %ld\n", pid);

printf("\n[%ld] : %s\n", rcvq.pid, rcvq.mbuff);

printf("----- Enter The User(PID) to send to : ");

}

}

}

//위와 마찬가지로 잡다한 선언 부분은 제외하고 설명하겠습니다.

int main()

{

while (1) {

printf("----- Notice : 0 = BroadCast Chat / PID = Send to User who is PID\n");

printf("----- Enter The User(PID) to send to : ");

gets(dpid);

Dest = atoi(dpid);

//메시지를 보내고싶은 곳을 확인합니다.

if (Dest == 0)

//0이라면 공유메모리를 사용하여 넘겨주게됩니다. 그러면 서버는 로그에 임시 채팅을 저장하게되고, 클라이언트도 확인할 수 있다.

{

sndq.type = GLOBAL\_TYPE;

printf("[Message] : ");

gets(message);

if (!strcmp(message, "EXIT"))

{

printf("Exit Chat...\n");

strcpy(sndq.mbuff, message);

msgsnd(MQID, &sndq, msgSize, 0);

shmdt(logmem);

return 0;

}

sprintf(logbuff, "[%ld] : %s\n",pid, message);

memcpy(tmpmem, logbuff, TEMP\_SIZE);

strcpy(sndq.mbuff, message);

msgsnd(MQID, &sndq, msgSize, 0);

}

else

{

sndq.type = Dest;

printf("[Message] : ");

gets(message);

if (!strcmp(message, "EXIT"))

{

printf("Exit Chat...\n");

strcpy(sndq.mbuff, message);

msgsnd(MQID, &sndq, msgSize, 0);

shmdt(logmem);

return 0;

}

strcpy(sndq.mbuff, message);

msgsnd(MQID, &sndq, msgSize, 0);

}

}

return 0;

}